INTELIGENCIA ARTIFICIAL

Combinando comportamientos reactivos y deliberativos

Álvaro Fernández García

Variables privadas del agente

Además de las explicadas en la práctica anterior, se han incoporado las siguientes:

- `struct estado` : Representa un estado para la planificación, contiene la fila, columna y orientación en la que se puede encontrar el agente.
- `struct nodo` : Representa el nodo que se utilizará para el grafo de estados en la búsqueda, contiene el estado que representa el nodo, el valor de la heurística de dicho nodo, y la secuencia de acciones para llegar a él desde el estado inicial.
- `reyes` : se trata de un vector de `pair<int,int>` que contiene la posición en el mapa de los reyes que nos hemos ido encontrando, esto se usará para que, una vez que tengamos el regalo, planificar una ruta hasta uno de los reyes *(el más cercano a nuestra posición)*
- `deliberar_en` : nos indica cuando el agente puede intentar planificar de nuevo, su valor es de 90, el cual se ha medido experimentalmente. Se ha seleccionado aquel número que presenta una mayor media de misiones cumplidas y el que también muestra un porcentaje de mapa explorado bastante bueno.
- `deliberando` : estará activo si el agente está ejecutando un plan y desactivado en caso contrario.
- `error_plan` : estará activo si el agente detecta que el plan puede fallar, para asi desactivar la variable `deliberando` y pasar a modo reactivo.
- `plan` : contiene el plan *(secuencia de acciones que se están ejecutando)*

(Se han eliminado las variables `girar_en` y `num_avances` utilizadas para ir de forma reactiva hasta un punto PK, ya que en esta práctica se hará de forma deliberativa)

Funciones auxiliares del agente

Además de las explicadas en la práctica anterior, se han incoporado las siguientes:

- `ObstaculoDelante` : determina si el agente tiene un obstáculo delante.
- `pathFinding_simple` : es un método para elaborar planes sencillos, se utiliza cuando la posición objetivo está dentro de los sensores del agente, y calcula la ruta en función de la misma. En la práctica se utilizará para seguir una ruta hasta un punto PK cercano si no estoy bien situado.
- `pathFinding` : es una implementación del método de búsqueda en grafos de escalada por máxima pendiente. Se utiliza para encontrar una ruta hasta un regalo para recogerlo y también para encontrar una ruta hasta un rey para entregar un regalo. Se asegura que encuentra un solución (mirar la función `calculaHeuristica`), y tiene un nivel máximo de búsqueda de 200 (*la máxima distancia a la que se pueden encontrar dos puntos*)
- `ImprimePlan` : Muestra en la teminal el plan establecido.
- `PuedoRecoger` Indica si puedo recoger un objeto o no. Se ha variado la política de mantener los objetos en mochila. Siempre se reserva un hueco para los regalos, el bikini y las zapatillas, considerados objetos más importantes, y luego se recogerá o bien un hueso o una llave, que se consideran objetos secundarios, ya que los lobos pueden esquivarse y abrir puertas es complicado. De esta forma garantizamos que no se superan los 4 objetos máximos que puede haber en la mochila. Devuelve un `pair<bool,int>` que indican la posición en la mochila auxiliar y si puedo recogerlo o no. Si no puede recogerse un objeto,

simplemente se esquivará.
- `calculaHeurística` : Calcula la heurística de un nodo. Para ello utiliza la distancia en filas y columnas desde el estado del nodo actual hasta el estado objetivo. También tiene en cuenta la orientación, por ejemplo, si quiero ir al Este, tendrá peor heurística un nodo en el que el agente mire al Oeste, que uno que mire al Norte, y ese a su vez será peor que un nodo que mire al Este. De esta forma se ha logrado una función monótona decreciente que nos evita los óptimos locales.
- `expandeNodo` : genera un descendiente de un nodo aplicándole una determinada acción.
- `guardaRey` : se trata de una función que recorre el vector de sensores para comprobar si cerca hay algún rey. A continuación, en función de la posición que ocupa dentro de los mismos, se calula su posición en filas y columnas, se comprueba si ese rey ya estaba en el vector `reyes` y si no es así se añade. Sólo se guardarán las posiciones de los reyes si el jugador está bien siuado. Esta operación se realiza en *Actualizar memoria*.

El método Think

Para realizar esta práctica se ha organizado el metodo `think` en distintas secciones, cada una de ellas correspondiente a acciones o procedimientos relacionados entre sí. Podemos distinguir las siguientes:

```
Comprobar si se ha reiniciado el juego;
-----------------------------------------------------------------------------------------
Actualizar posición;
-----------------------------------------------------------------------------------------
Actualizar memoria:
    > Completar el mapa;
    > Si estoy bien situado, compruebo si hay un rey cerca y guardo su posición;
-----------------------------------------------------------------------------------------
Elaboración y comprobación de planes Deliberativos: Planificación;
    > Comprobar la validez del plan actual en caso de existir;
    > Planificar en caso de que sea necesario y factible;
-----------------------------------------------------------------------------------------
Acciones que NO implican movimiento;
-----------------------------------------------------------------------------------------
Acciones que implican movimiento;
-----------------------------------------------------------------------------------------
Ajustes finales;
-----------------------------------------------------------------------------------------
Retornar valores;
```

Comportamiento Reactivo

El comportamiento reactivo está localizado fundamentalmente en las zonas de *Acciones que no implican movimiento* y *Acciones que implican movimiento*. Esta es una de las principales modificaciones que se han introducido con respecto a la práctica anterior, ya que antes no estaban clasificadas de dicha forma. Entre las *acciones que no implican movimiento* encontramos la recogida de objetos, buscar objetos en la mochila, equiparlos cuando sea necesario, abrir puertas y darle huesos a los lobos así como los regalos a los reyes.

Por otra parte entre las *acciones que implican movimiento* encontramos la ejecución de un plan si este está activo, o seguir el comportamiento de movimiento por defecto (andar si delante hay terreno del tipo **T**, **S**, si hay agua y tengo el bikini, si hay bosque y tengo las zapatillas, girar…)

Comportamiento Deliberativo: Planificación

Se estructura en dos partes fundamentales:

Comprobar validez del plan: Su estructura es la siguiente:

- Si se produjo un error en el plan dejo de ejecutarlo.
- Si estoy ejecutando un plan, tengo que avanzar, pero delante hay un obstáculo, compruebo de que obstaculo se trata:
  - Se trata de un aldeano o un lobo, me espero, el plan no falla.
  - Es un objeto y puedo recogerlo, el plan no falla, si no puedo recogerlo el plan falla.
  - Si es agua, bosque o una puerta, tengo el objeto correspondiente y puedo equiparmelo, el plan no falla.
- En cualquier otro caso el plan falla.

Planificar: Se pueden realizar 3 planes distintos:

- Si hay un punto PK cerca, no estoy bien situado y no tengo otro plan, busco un plan para llegar a él *(esto es una modificación con respecto a la práctica anterior, que lo hacia de forma reactiva)* Para ello utiliza el método `pathFinding_simple`.
- Si estoy bien situado, no tengo regalo, hay regalos para recoger, no estoy ejecutando un plan y puedo planificar, busco un plan para llegar hasta el regalo. Para ello utiliza el método `pathFinding`, el cual aplicará la escalada por máxima pendiente y la heurística.
- Si estoy bien situado, tengo regalo, conozco la posición de algún rey, no estoy ejecutando un plan y puedo planificar, busco un plan para llegar hasta el rey y darle el regalo. Para ello utiliza el método `pathFinding`, el cual aplicará la escalada por máxima pendiente y la heurística. Se busca un rey que este lo más cerca posible del jugador.

Conmutación entre reactivo y deliberativo

La forma en la que se alternan ambos comportamientos es muy sencilla. Siempre que pueda (*es decir, se den las condiciones para ello*), el agente va a intentar establecer un plan. Comenzará a ejecutarlo, si lo consigue, seguirá planificando y ejecutando planes, sin embargo, si se produce un error (*fallo en el plan*) mientras se encontraba ejecutando alguno de ellos, el agente no intentará volver a planificar hasta que no realice 90 acciones en comportamiento reactivo (el valor que nos indica la variable `deliberar_en`) de ahí que hablemos en apartados anteriores de *"puedo planificar"*. Además, si no puede fijar ningún plan porque no se cumple ninguna condición para ello, continuará en modo reactivo hasta que alguna condición sea verdadera.

Resultados

Aquí se muestran los resultados obtenidos para este comportamiento:

Tamaño Mapa	% explorado	Misiones Cumplidas
30	99.8889	17
50	100	35
100	96.36	5
100n	99.36	12
100n2	79.19	2